Sample-Hold Timer

IP177 board interrupt

Thu, May 1, 2008

Support for up to 4 IndustryPack (IP) boards is available for the MVME-162 CPU board, which are interfaced via the IndustryPack Interface Controller (IPIC) chip on that board. This note describes a simple support for interrupts, in which a Bit is ultimately set as a result of the occurrence of such an interrupt. The logic is included in local application SHIN.

The original motivation for this support was for the Main Injector HLRF systems, in which an interrupt is used to denote when to sample the phase detector error signal, so that it can be used for reducing the phase error. The regulation logic is furnished by MRF0, which adjusts a D/A to minimize the phase detector error. This LA monitors a Bit to know when (on what 15 Hz cycle) to sample the error signal and make any adjustment. The actual interrupt-related logic is handled by SHIN, who sets the same Bit if the interrupt occurs.

The parameters for MRF0 in node06D1 are as follows, sampled from Page E:

```
ENABLE  B<00B6>*MRF0 ENABLE
RF ON   B<0176> GAP ENV ABV THRS
SHTRIG  B<00CD> SHIN S&H STATUS
PDERR   C<0100> PD01E  1.642 DEG
FBOFFS  C<011D> FB01O -1.724 V
EVT MSK C<0050> RF01AT 0
```

The corresponding parameters for SHIN in the same front end are:

```
ENABLE  B<00CC>*SHIN ENABLE
STATUS  B<00CD> SHIN S&H STATUS
TREGS HI <FFF5>
TREGS LO <8200>
COUNTER C<000C> SHM01C-13187
```

Here, SHIN sets Bit 00CD when it sees that the timer interrupt occurred, and MRF0 samples Bit 00CD to know when to sample the phase detector error signal. The address of the timer IP board in this example is 0xFFF58200, which means it is in slot c, as specified by the digit 2. (Slots a, b, c, d, are indicated by 0, 1, 2, 3, respectively.)

During its initialization, SHIN calls INZSHINT to enable the interrupt, and it also looks up the byte address in the BADDR table corresponding to the given Bit. Then, each cycle, it either sets the indicated bit in that byte, or it clears it, based upon whether the word counter that the interrupt routine increments has changed, or not. On the next cycle, when the Data Access Table is processed, the BBYTE table, which houses the digital data pool, will be updated with the latest value of this byte. Then, if MRF0 sees that the bit is set, it will do its regulation.

What does INZSHINT do? It needs the slot number 0–3, and it needs to know whether the IRQ0 or IRQ1 interrupt from the IP board in that slot is to be used. It obtains the slot number from the timer board address. It obtains the IRQ0/IRQ1 selection from the least significant bit of the timer board address. (In the example above, if IRQ1 were to be used, the timer board address would be entered as 0xFFF58201. The actual board base address always ends in 8 bits of zero, and INZSHINT knows it.)

Once the slot number and IRQ0/IRQ1 option are known, INZSHINT uses the interrupt vector number (range 0x68–0x6F) to look up the exception vector address in low memory. It saves

what is there, then sets the exception vector to the address of the corresponding interrupt routine. It then enables the interrupt in the corresponding IPIC interrupt control register, setting the interrupt level to 1. Next it saves the address of the caller's word counter so that the interrupt routine can find it.

On the timer board, it sets the interrupt vector number for either the `IRQ0` or `IRQ1` interrupt into the appropriate register, in the low byte of the word at offset `0x5A` or `0x5C`. Finally, it enables the corresponding interrupt in the IRQ Vector Enable register, in the low byte of the word at offset `0x5E`. At this point, everything is ready to receive a timer interrupt.

When the interrupt occurs, the interrupt routine looks up its counter word address, and it increments the counter word. It then exits. There is no need to clear the interrupt, since IPIC handles that automatically when the CPU recognizes it.

The next time that `SHIN` gets a `cycle` call, it sees that the counter word value has changed, so it sets the selected Bit. (If it has not changed, it clears the Bit.) It also writes the new counter value to the given Chan, if any, as a diagnostic. (One can thus see this incrementing count on Page `E`.) As mentioned above, it is the selected Bit value that `MRF0` monitors.

If the `SHIN` instance is disabled, the interrupt is also disabled via `CANSHINT`, both in the IPIC and in the timer board, and the exception vector in low memory is restored to its saved value.

This latest version of `SHIN` allows for more than one instance. Each of up-to-8 IP board interrupts can be handled, specified via the timer board address parameter, where the least significant bit of the address selects whether `IRQ0` or `IRQ1` in that board is to be used.

Two arrays of 8 addresses are needed to keep track of each possible word counter and the previous copy of the exception vector. The latter is needed in case the instance is disabled. These two arrays are allocated in 64 bytes of low memory based at at `0x000CC0`. The array of counter word pointers is followed by the array of exception vectors.

For testing, here are some relevant addresses (for slot c) to watch to confirm `SHIN` behavior:

`FFF58258` Timer board spare, `IRQ0` Vector, `IRQ1` Vector, IRQ Vector Enable (8 bytes)

`FFFBC010` IPIC Interrupt Control Registers (two bytes for each slot)

`000001B0` Exception vectors for vectors `0x6C`, `0x6D`, as used for slot c

`00000CD0` Two word counter addresses for slot c

`00000CF0` Two saved exception vectors for slot c

Note that the counter channel parameter for `SHIN` is optional. If that parameter is zero, then `SHIN` ignores it. If it is nonzero, `SHIN` provides a copy of the word counter via that channel.

Finally, when any `SHIN` instance parameter is changed, but for the optional counter channel, one must restart the `SHIN` instance in order for the change to take effect.